Tree-Like Resolution Is Superpolynomially Slower Than DAG-Like Resolution for the Pigeonhole Principle

AUTHOR(S):

Iwama, Kazuo; Miyazaki, Shuichi

Tree-Like Resolution Is Superpolynomially Slower Than DAG-Like Resolution for the Pigeonhole Principle

Kazuo Iwama* and Shuichi Miyazaki

School of Informatics, Kyoto University, Kyoto 606-8501, Japan
{iwama, shuichi}@kuis.kyoto-u.ac.jp

Abstract. Our main result shows that a shortest proof size of tree-like resolution for the pigeonhole principle is superpolynomially larger than that of DAG-like resolution. In the proof of a lower bound, we exploit a relationship between tree-like resolution and backtracking, which has long been recognized in this field but not been used before to give explicit results.

1 Introduction

A *proof system* is a nondeterministic procedure to prove the unsatisfiability of CNF formulas, which proceeds by applying (usually simple) rules each of which can be computed in polynomial time. Therefore, if there is a proof system which runs in a polynomial number of steps for every formula, then NP=coNP [5]. Since this is not likely, it has long been an attractive research topic to find exponential lower bounds for existing proof systems. There are still a number of well-known proof systems for which no exponential lower bounds have been found, such as Frege systems [3].

Resolution is one of the most popular and simplest proof systems. Even so, it took more than two decades before Haken [6] finally obtained an exponential lower bound for the pigeonhole principle. This settlement of the major open question, however, has stimulated continued research on the topic [1, 4, 8]. The reason is that Haken's lower bound is quite far from being tight and his proof, although based on an excellent idea later called *bottleneck counting*, is not so easy to read.

Tree-like resolution is a restricted resolution whose proof must be given as not directed acyclic graph (DAG) but a tree. It is a common perception that tree-like proof systems are exponentially weaker than their DAG counterparts. Again, however, proving this for resolution was not easy: In [2], Bonet et al. showed that there exists a formula whose tree-like resolution requires $2^{\Omega(n^\epsilon)}$ steps for some ϵ, while $n^{O(1)}$ steps suffice for DAG-like resolution.

In this paper, we give such a separation between tree-like and DAG-like resolutions using the pigeonhole principle that is apparently the most famous

* Supported in part by Scientific Research Grant, Ministry of Japan, 10558044, 09480055 and 10205215.

and well-studied formula. Our new lower bound for tree-like resolution is $(\frac{n}{4})^{\frac{n}{4}}$ steps for the $n+1$ by n pigeonhole formula. The best previous lower bound is 2^n [4] which is not enough for such a (superpolynomial) separation since the best known upper bound of DAG-like resolution is $O(n^3 2^n)$ [4]. Our new lower bound shows that tree-like resolution is superpolynomially slower than DAG-like resolution for the pigeonhole principle.

Another contribution of this paper is that the new bound is obtained by fully exploiting the relationship between resolution and backtracking. This relationship has long been recognized in the community, but it was informal and no explicit research results have been reported. This paper is the first to formally claim a benefit of using this relationship. Our lower bound proof is completely based on backtracking, whose top-down structure makes the argument surprisingly simple and easy to follow.

In Sec. 2, we give basic definitions and notations of resolution, backtracking and the pigeonhole principle. We also show the relationship between resolution and backtracking. In Sec. 3, we prove an $(\frac{n}{4})^{\frac{n}{4}}$ lower bound of tree-like resolution for the pigeonhole principle. In Sec. 4, we give an upper bound $O(n^2 2^n)$ of the DAG-like resolution which is slightly better than $O(n^3 2^n)$ proved in [4]. It should be noted that our argument in this paper holds also for a generalized pigeonhole principle, called the *weak pigeonhole principle*, which is an m by n $(m > n)$ version of the pigeonhole principle. Finally, in Sec. 5, we mention future research topics related to this paper.

2 Preliminaries

A *variable* is a logic variable which takes the value *true* (1) or *false* (0). A *literal* is a variable x or its negation $\overline{x}$. A *clause* is a sum of literals and a *CNF formula* is a product of clauses. A *truth assignment* for a CNF formula f is a mapping from the set of variables in f into $\{0,1\}$. If there is no truth assignment that satisfies f, we say that f is *unsatisfiable*.

The pigeonhole principle is a tautology which states that there is no bijection from a set of $n+1$ elements into a set of n elements. PHP_n^{n+1} is a CNF formula that expresses a negation of the pigeonhole principle; hence PHP_n^{n+1} is unsatisfiable. PHP_n^{n+1} consists of $n(n+1)$ variables $x_{i,j}$ $(1 \le i \le n+1, 1 \le j \le n)$, and $x_{i,j} = 1$ means that i is mapped to j. There are two sets of clauses. The first part consists of clauses $(x_{i,1} + x_{i,2} + \cdots + x_{i,n})$ for $1 \le i \le n+1$. The second part consists of clauses $(\overline{x_{i,k}} + \overline{x_{j,k}})$, where $1 \le k \le n$ and $1 \le i < j \le n+1$. Thus there are $(n+1) + \frac{1}{2}(n^2(n+1))$ clauses in total.

Resolution is a proof system for unsatisfiable CNF formulas. It consists of only one rule called an *inference rule*, which infers a clause $(A+B)$ from two clauses $(A+x)$ and $(B+\overline{x})$, where each of A and B denotes a sum of literals such that there is no variable y that appears positively (negatively, resp.) in A and negatively (positively, resp.) in B. We say that the variable x is *deleted* by this inference. A *resolution refutation* for f is a sequence of clauses $C_1, C_2, \cdots, C_t$, where each C_i is a clause in f or a clause inferred from clauses C_j and C_k $(j,k < i)$, and the last clause C_t is the empty clause $(\emptyset)$. The size of a resolution refutation is the number of clauses in the sequence.

A resolution refutation can be represented by a directed acyclic graph. (In this sense, we sometimes use the term *DAG-like resolution* instead of "resolution.") If the graph is restricted to a tree, namely, if we can use each clause only once to infer other clauses, then the refutation is called *tree-like resolution refutation* and the tree is called a *resolution refutation tree* (*rrt*). More formally, an rrt for a formula f is a binary rooted tree. Each vertex v of an rrt corresponds to a clause, which we denote by $Cl(v)$. $Cl(v)$ must satisfy the following conditions: For each leaf v, $Cl(v)$ is a clause in f, and for each vertex v other than leaves, $Cl(v)$ is a clause inferred from $Cl(v_1)$ and $Cl(v_2)$, where v_1 and v_2 are v's children. For the root v, $Cl(v)$ is the empty clause ($\emptyset$). The size of an rrt is the number of vertices in the rrt.

Backtracking (e.g., [7]) determines whether a given CNF formula is satisfiable or not in the following way: For a formula f, variable x and $a \in \{0,1\}$, let $f_{x=a}$ be the formula obtained from f by substituting value a for x. In each step, we choose a variable x and calculate $f_{x=0}$ and $f_{x=1}$ recursively. If f is simplified to the constant true function at any point, then f is satisfiable, and otherwise, f is unsatisfiable.

Backtracking search is also represented by a tree. A *backtracking tree* (*btt*) for an unsatisfiable formula f is a binary rooted tree satisfying the following three conditions: (i) Two edges e_1 and e_2 from a vertex v are labeled as $(x = 0)$ and $(x = 1)$ for a variable x. (ii) For each leaf v and each variable x, x appears at most once (in the form of $(x = 0)$ or $(x = 1)$) in the path from the root to v. (iii) For each vertex v, let $As(v)$ be the (partial) truth assignment obtained by collecting labels of edges in the path from the root to v. Then, for each v, v is a leaf iff f becomes false by $As(v)$. (Recall that we consider only unsatisfiable formulas.) The size of a btt is the number of vertices in the btt.

Proposition 1. *Let f be an unsatisfiable CNF formula. If there exists an rrt for f whose size is k, then there exists a btt for f whose size is at most k.*

Proof. Let R be an rrt for f. It is known that a shortest tree-like resolution refutation is *regular*, i.e., for each path from the root to a leaf, each variable is deleted at most once [9]. Thus we can assume, without loss of generality, that R is regular.

From R, we construct a btt B which is isomorphic to R. What we actually do is to give a label to each edge in the following way: Let v_i be a vertex of R and let v_{i_1} and v_{i_2} be its children. Suppose $Cl(v_i) = (A + B)$, $Cl(v_{i_1}) = (A + x)$ and $Cl(v_{i_2}) = (B + \overline{x})$. Then the labels $(x = 0)$ and $(x = 1)$ are assigned to edges (u_i, u_{i_1}) and (u_i, u_{i_2}), respectively, where u_i is a vertex of B corresponding to v_i of R. We shall show that this B is a btt for f.

It is not hard to see that the conditions (i) and (ii) for btt are satisfied. In the following, we show that for any leaf u of B, f becomes false by $As(u)$. This is enough for the condition (iii) because if f becomes false in some non-leaf node, then we can simply cut the tree at that point and can get a smaller one. To this end, we prove the following statement by induction: For each i, the clause $Cl(v_i)$ of R becomes false by the partial assignment $As(u_i)$ of B. For the induction basis, it is not hard to see that the statement is true for the root. For the induction hypothesis, suppose that the statement is true for a vertex

v_i, i.e., $Cl(v_i)$ becomes false by $As(u_i)$. Now we show that the statement is also true for v_i's children. Let v_{i_1} and v_{i_2} be v_i's children, and let $Cl(v_i) = (A + B)$, $Cl(v_{i_1}) = (A + x)$ and $Cl(v_{i_2}) = (B + \overline{x})$. Then the label of the edge (v_i, v_{i_1}) is $(x = 0)$, and hence, $As(u_{i_1}) = As(u_i) \cup \{(x = 0)\}$. Since $As(u_i)$ makes $(A + B)$ false, $As(u_{i_1})$ makes $(A+x)$ false. The same argument shows that $As(u_{i_2})$ makes $Cl(v_{i_2})$ false. Now the above statement is proved, which immediately implies that $As(u_i)$ makes f false for every leaf u_i of B. □

Thus to show a lower bound of tree-like resolution, it suffices to show a lower bound on the size of btts.

3 A Lower Bound

In this section, we prove a lower bound on the size of tree-like resolution for PHP_n^{n+1}.

Theorem 1. *Any tree-like resolution refutation for* PHP_n^{n+1} *requires* $(\frac{n}{4})^{\frac{n}{4}}$ *steps.*

Proof. By Proposition 1, it is enough to show that any btt for PHP_n^{n+1} requires $(\frac{n}{4})^{\frac{n}{4}}$ vertices. For simplicity, we consider the case that n is a multiple of 4. Let B be an arbitrary btt for PHP_n^{n+1}. As we have seen before, each vertex v of B corresponds to a partial truth assignment $As(v)$. For a better exposition, we use an $n+1$ by n array representation to express a partial assignment for PHP_n^{n+1}. Fig. 1 shows an example of PHP_4^5. A cell in column i and row j corresponds to the variable $x_{i,j}$. We consider that the value 1 (resp. 0) is assigned to the variable $x_{i,j}$ if the (i, j) entry of the array is 1 (resp. 0). For example, Fig. 1 (b) expresses a partial assignment such that $x_{1,4} = x_{2,2} = x_{4,4} = x_{5,3} = 0$, $x_{3,3} = x_{4,1} = 1$. It should be noted that PHP_n^{n+1} becomes false at a vertex v iff (i) $As(v)$ contains a column filled with 0s or (ii) $As(v)$ contains a row in which two 1s exist.

Here are some notations. For a partial assignment A, a 0 in the (i, j) entry of A is called a *bad* 0 if neither the column i nor the row j contains a 1, and is called a *good* 0 otherwise. (The reason why we use terms "bad" and "good" will be seen later. Bad 0s make it difficult to count the number of vertices in B in our analysis.) $\#BZ(A)$ denotes the number of bad 0s in A. A variable $x_{i,j}$ is called an *active variable* for A if $x_{i,j}$ is not yet assigned (that is, (i, j) entry of A is blank) and neither the column i nor the row j contains a 1. For example, let A_0 be the assignment in Fig. 1 (b). Then $\#BZ(A_0) = 2$ (0s assigned to $x_{1,4}$ and $x_{2,2}$). For example, $x_{2,4}$ is an active variable for A_0. Let v be a vertex of B and v_0 and v_1 be its children. Suppose that labels of edges (v, v_0) and (v, v_1) are $(x = 0)$ and $(x = 1)$, respectively. Then we write $Var(v) = x$, namely, $Var(v)$ denotes the variable selected for substitution at the vertex v. We call v_0 a *false-child* of v and write $F(v) = v_0$. Similarly we call v_1 a *true-child* of v and write $T(v) = v_1$.

We want to show a lower bound on the number of vertices in B. To this end, we construct a tree S from B. Before showing how to construct S, we show some properties of S: The set of vertices of S is a subset of the set of vertices of B, so the number of vertices of S gives a lower bound on the number of vertices of B. Each internal vertex of S have either a single child or exactly $\frac{n}{4}$ children. The

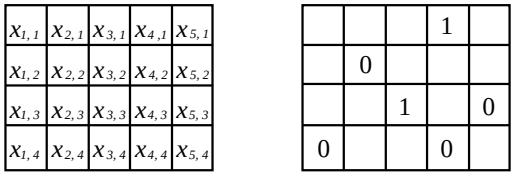

Fig. 1. An array representation of a partial assignment for PHP_4^5

height of S is $\frac{n}{2}$; more precisely, the length of any path from the root to a leaf is exactly $\frac{n}{2}$.

Now we show how to construct S. The root of S is the root of B. For each vertex v of S, we select the set $CH(v)$ of v's children in the following way: $CH(v)$ is initially empty and vertices are added to $CH(v)$ one by one while tracing the tree B down from the vertex v. We look at vertices $T(v), T(F(v)), T(F(F(v)))(= T(F^2(v))), \cdots, T(F^l(v)), \cdots$ in this order. We add $T(F^l(v))$ $(l \geq 0)$ to $CH(v)$ if $Var(F^l(v))$ is an active variable for $As(F^l(v))$. Fig. 2 illustrates how to trace the tree B when we construct the tree S. In this example, $T(F^2(v))$ is "skipped" because $Var(F^2(v))$ is not an active variable. We stop adding vertices if $|CH(v)|$ becomes $\frac{n}{4}$. In this case, v has exactly $\frac{n}{4}$ children.

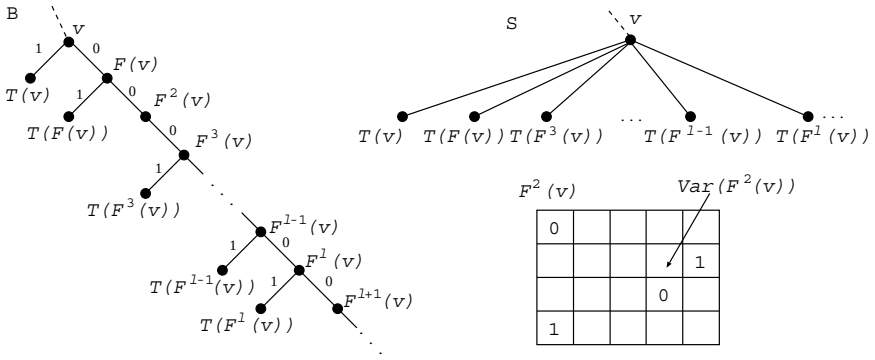

Fig. 2. A part of S constructed from B

However, there is one exceptional condition to stop adding vertices to $CH(v)$ even if $|CH(v)|$ is less than $\frac{n}{4}$; when the number of bad 0s in some column reaches $\frac{n}{2}$, we stop adding vertices to $CH(v)$. More formally, let us consider a vertex $F^l(v)$. Suppose that the number of bad 0s in each column of $As(F^l(v))$ is at most $\frac{n}{2} - 1$. Also, suppose that $Var(F^l(v))$ is $x_{i,j}$ where the column i of $As(F^l(v))$ contains exactly $\frac{n}{2} - 1$ bad 0s and there is no 1 in the row j of $As(F^l(v))$. (See Fig. 3 for an example of the case that $n = 12$. There are eight 0s in the column i. Among them, five 0s are bad 0s.) Then $As(F^{l+1}(v))$

contains $\frac{n}{2}$ bad 0s in the column i, and hence we do not look for vertices any more, namely, $T(F^l(v))$ is the last vertex added to $CH(v)$. (Note that $T(F^l(v))$ is always selected since $x_{i,j}$ is an active variable for $As(F^l(v))$.) In this case, $|CH(v)|$ may be less than $\frac{n}{4}$. If so, we adopt only the last vertex as a child of v, i.e., only $T(F^l(v))$ is a child of v in S. Thus, in this case, v has only one child. It should be noted that, in the tree S, every assignment corresponding to a vertex of depth i contains exactly i 1s. We continue this procedure until the length of every path from the root to a leaf becomes $\frac{n}{2}$.

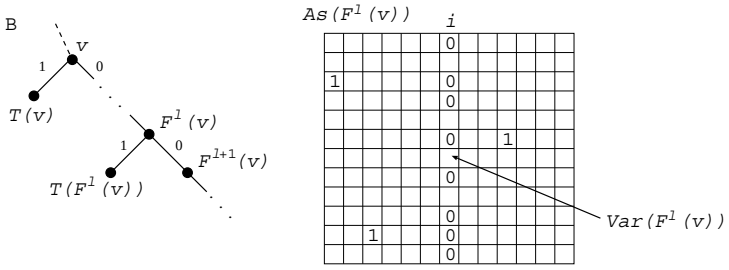

Fig. 3. A condition to stop tracing B

We then show that it is possible to construct such S from any B. To see this, it suffices to show that we never reach a leaf of B while tracing B to construct S. Recall the definition of a backtracking tree: For any leaf u of btt, $As(u)$ makes the CNF formula false. For $As(u)$ to make PHP_n^{n+1} false, $As(u)$ must have a row containing two 1s or a column full of 0s. The former case does not happen in S because we have skipped such vertices in constructing S. The latter case does not happen for the following reason: Recall that once the number of bad 0s in some column reaches $\frac{n}{2}$, we stop tracing the tree B. So, as long as we trace B in constructing S, we never visit an assignment such that the number of bad 0s in a column exceeds $\frac{n}{2} - 1$. Also, recall that the number of 1s in $As(v)$ is at most $\frac{n}{2}$ since v's depth in S is at most $\frac{n}{2}$. So the number of good 0s in a column is at most $\frac{n}{2}$. Hence the number of 0s in each column is at most $n - 1$, and so, no column ever becomes filled with 0s. Now let us consider the following observation which helps to prove later lemmas.

Observation 1. Consider a vertex v in B and let $v' = F^l(v)$ for some l (see Fig. 4). Suppose first that $T(v')$ is added to $CH(v)$ in constructing S. Then $Var(v')$ must be an active variable for v', and hence $\#BZ(As(F(v'))) = \#BZ(As(v')) + 1$. On the other hand, suppose that $T(v')$ is not added to $CH(v)$ in constructing S. Then $Var(v')$ is not an active variable for v'. Therefore, $\#BZ(As(F(v'))) = \#BZ(As(v'))$.

Now we have a tree S having the following properties: The length of any path from the root to a leaf is $\frac{n}{2}$. Every vertex in S except for leaves has exactly $\frac{n}{4}$ children or one child. When a vertex v has one child, we call the edge between v and its child a *singleton*.

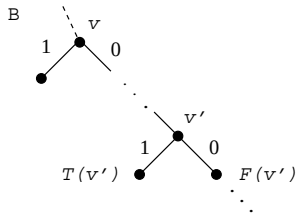

Fig. 4. An example for Observation 1

Lemma 1. *Consider an arbitrary path* $P = u_0u_1u_2\cdots u_{\frac{n}{2}}$ *in* S*, where* u_0 *is the root and* $u_{\frac{n}{2}}$ *is a leaf. For each* k*, if* (u_{k-1}, u_k) *is not a singleton, then* $\#BZ(As(u_k)) \leq \#BZ(As(u_{k-1})) + \frac{n}{4} - 1$.

Proof. Consider u_{k-1} and u_k in the path P. Let v be the vertex in the btt B corresponding to u_{k-1}. Then there is some l such that $T(F^l(v))$ corresponds to u_k. By Observation 1, $\#BZ(As(F^l(v))) - \#BZ(As(v))$ is equal to the number of vertices added to $CH(v)$ among $T(v), T(F(v)), T(F^2(v)), \cdots, T(F^{l-1}(v))$. So $\#BZ(As(F^l(v))) - \#BZ(As(v)) \leq \frac{n}{4} - 1$. Note that $As(T(F^l(v)))$ is the result of adding one 1 to some active variable of $As(F^l(v))$, so $\#BZ(As(T(F^l(v)))) \leq \#BZ(As(F^l(v)))$. Hence $\#BZ(As(T(F^l(v)))) \leq \#BZ(As(v)) + \frac{n}{4} - 1$, namely, $\#BZ(As(u_k)) \leq \#BZ(As(u_{k-1})) + \frac{n}{4} - 1$. □

Lemma 2. *Consider an arbitrary path* $P = u_0u_1u_2\cdots u_{\frac{n}{2}}$ *in* S*, where* u_0 *is the root and* $u_{\frac{n}{2}}$ *is a leaf. For each* k*, if* (u_{k-1}, u_k) *is a singleton, then* $\#BZ(As(u_k)) \leq \#BZ(As(u_{k-1})) - \frac{n}{4}$.

Proof. Suppose that the edge (u_{k-1}, u_k) is a singleton. Let v and $T(F^l(v))$ be vertices in B corresponding to u_{k-1} and u_k, respectively. The same argument as in Lemma 1 shows that $\#BZ(As(F^l(v))) - \#BZ(As(v)) \leq \frac{n}{4} - 1$. Let $Var(F^l(v)) = x_{i,j}$. Since (u_{k-1}, u_k) is a singleton, there are $\frac{n}{2} - 1$ bad 0s in the column i of $As(F^l(v))$. Thus substituting the value 1 for the variable $x_{i,j}$ makes at least $\frac{n}{2} - 1$ bad 0s good, namely, $\#BZ(As(T(F^l))) \leq \#BZ(As(F^l(v))) - (\frac{n}{2} - 1)$. Hence $\#BZ(As(T(F^l(v)))) \leq \#BZ(As(v)) - \frac{n}{4}$, namely, $\#BZ(As(u_k)) \leq \#BZ(As(u_{k-1})) - \frac{n}{4}$. □

Lemma 3. *Consider an arbitrary path* $P = u_0u_1u_2\cdots u_{\frac{n}{2}}$ *in* S*, where* u_0 *is the root and* $u_{\frac{n}{2}}$ *is a leaf. The number of singletons in* P *is at most* $\frac{n}{4}$.

Proof. Suppose that there exist more than $\frac{n}{4}$ singletons in the path P. We count the number of bad 0s of assignments along P. At the root, $\#BZ(As(u_0)) = 0$. Going down the path from the root u_0 to the leaf $u_{\frac{n}{2}}$, the number of bad 0s is increased at most $(\frac{n}{4} - 1)(\frac{n}{4} - 1) < \frac{n^2}{16}$ by Lemma 1, and is decreased at least $\frac{n}{4}(\frac{n}{4} + 1) > \frac{n^2}{16}$ by Lemma 2. This is a contradiction because the number of bad 0s becomes negative at the leaf. This completes the proof. □

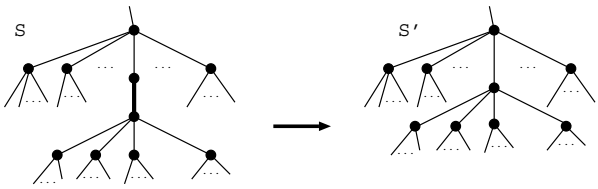

Fig. 5. Shrinking S to obtain S'

Finally we "shrink" the tree S by deleting all singletons from S. Let S' be the resulting tree (see Fig. 5). Each vertex of S' has exactly $\frac{n}{4}$ children and the length of every path from the root to a leaf is at least $\frac{n}{4}$ by Lemma 3. So there are at least $(\frac{n}{4})^{\frac{n}{4}}$ vertices in S', and hence, the theorem follows. □

Remark. By slightly modifying the above proof, we can get a better lower bound of $(\frac{n}{4\log^2 n})^n$. For the tree S in the above proof, we restrict the number of children of each node, the maximum number of bad 0s in each column, and the height of the tree with $\frac{n}{4}$, $\frac{n}{2}$, and $\frac{n}{2}$, respectively. To get a better lower bound, we let them be $\delta^2 n$, δn, and $(1-\delta)n$, respectively, with $\delta = \frac{1}{\log n}$. Then, the number of singletons in each path is at most $\delta(1-\delta)n$ and hence we can obtain a lower bound $(\delta^2 n)^{(1-\delta)^2 n} \geq (\frac{n}{4\log^2 n})^n$.

4 An Upper Bound

It is known that the size of a DAG-like resolution refutation for PHP_n^{n+1} is $O(n^3 2^n)$[4]. Here we show a slightly better upper bound which is obtained by the similar argument as [4]. We can also obtain an upper bound of tree-like resolution refutation for PHP_n^{n+1} as a corollary.

Theorem 2. *There is a DAG-like resolution refutation for PHP_n^{n+1} whose size is $O(n^2 2^n)$.*

Proof. Let Q and R be subsets of $\{1, 2, \cdots, n+1\}$ and $\{1, 2, \cdots, n\}$, respectively. Then we denote by $P_{Q,R}$ the sum of positive literals $x_{i,j}$, where $i \in Q$ and $j \in R$. Let $[i, j]$ denote the set $\{i, i+1, \cdots, j-1, j\}$.

We first give a rough sketch of the refutation and then describe it in detail. The 0th level of the refutation has the single clause $P_{\{1\},[1,n]}$. The first level consists of n clauses $P_{[1,2],R^{(n-1)}}$ for all sets $R^{(n-1)} \subset [1, n]$ of size $n-1$. The second level consists of ${}_nC_{n-2}$ clauses $P_{[1,3],R^{(n-2)}}$ for all sets $R^{(n-2)} \subset [1, n]$ of size $n-2$. Generally speaking, the ith level consists of ${}_nC_{n-i}$ clauses $P_{[1,i+1],R^{(n-i)}}$ for all $R^{(n-i)} \subset [1, n]$ of size $n-i$. At the $(n-1)$th level, we have ${}_nC_1 = n$ clauses $P_{[1,n],\{1\}}, P_{[1,n],\{2\}}, \cdots, P_{[1,n],\{n\}}$. Finally, at the nth level, we have the empty clause. We call the clauses described here *main clauses*. Note that there are $\Sigma_{i=0}^{i=n}({}_nC_i) = 2^n$ main clauses. Fig. 6 shows an example of the case when $n = 4$. A "+" sign in the (i, j) entry means the existence of the literal $x_{i,j}$ in that clause.

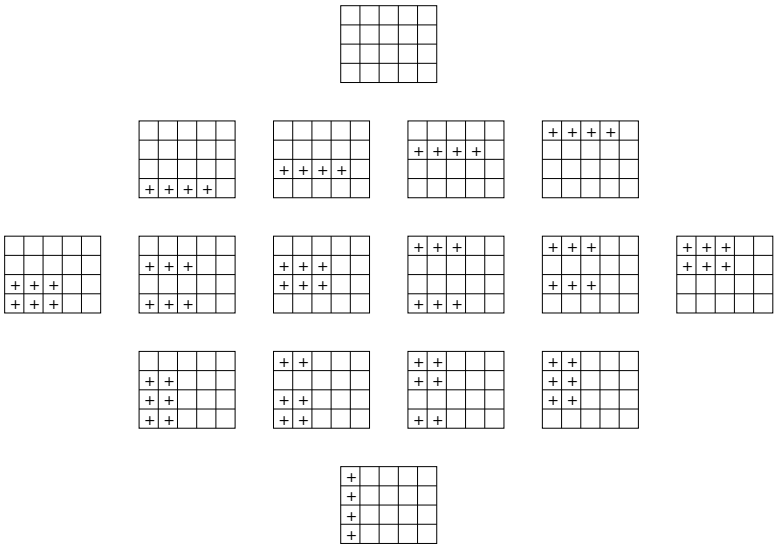

Fig. 6. Main clauses of the refutation

Then we describe the detail of the refutation. Each clause at the ith level is obtained by using i clauses of the $(i-1)$th level and some initial clauses. To construct a clause $P_{[1,i+1],\{j_1,j_2,\cdots,j_{n-i}\}}$ in the ith level, we use i clauses $P_{[1,i],\{j_1,j_2,\cdots,j_{n-i},k\}}$ for all $k \notin \{j_1, j_2, \cdots, j_{n-i}\}$. First, for each k, we construct a clause $P_{[1,i],\{j_1,j_2,\cdots,j_{n-i}\}} \cup \overline{x_{i+1,k}}$ using the clause $P_{[1,i],\{j_1,j_2,\cdots,j_{n-i},k\}}$ of the $(i-1)$th level and i clauses $(\overline{x_{1,k}} + \overline{x_{i+1,k}})(\overline{x_{2,k}} + \overline{x_{i+1,k}}) \cdots (\overline{x_{i,k}} + \overline{x_{i+1,k}})$. Then we construct a target clause $P_{[1,i+1],\{j_1,j_2,\cdots,j_{n-i}\}}$ by using those i clauses $P_{[1,i],\{j_1,j_2,\cdots,j_{n-i}\}} \cup \overline{x_{i+1,k}}$ and the initial clause $P_{\{i+1\},[1,n]}$. Fig. 7 illustrates an example of deriving $P_{[1,3],\{1,4\}}$ in the second level from clauses $P_{[1,2],\{1,2,4\}}$ and $P_{[1,2],\{1,3,4\}}$ in the first level. Similarly as the "+" sign, a "−" sign in the (i,j) entry means the existence of the literal $\overline{x_{i,j}}$ in that clause.

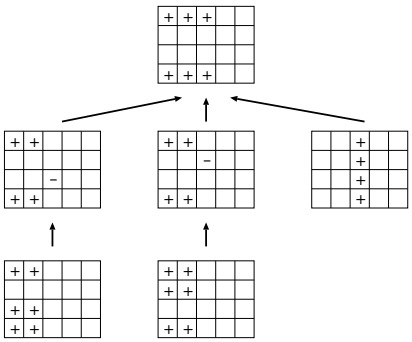

Fig. 7. Constructing a main clause

Thus to construct each main clause, we need $O(n^2)$ steps. Since there are 2^n main clauses, the size of the above refutation is bounded by $O(n^2 2^n)$. □

Corollary 1. *There is a tree-like resolution refutation for PHP_n^{n+1} whose size is $O(n^2 n!)$.*

Proof. This is obtained by reforming the directed acyclic graph of the refutation obtained in Theorem 2 into a tree in a trivial manner. For main clauses, we have one level-n clause, n level-$(n-1)$ clauses, $n(n-1)$ level-$(n-2)$ clauses and so on. Generally speaking, we have $n(n-1)\cdots(i+1)$ level-i clauses. Thus we have $1 + \Sigma_{i=0}^{i=n-1} n(n-1)\cdots(i+1) \leq 2n!$ main clauses. Each main clause is constructed in $O(n^2)$ steps and hence the size of the refutation is $O(n^2 n!)$. □

5 Concluding Remarks

By Theorems 1 and 2, we can see that the size of any tree-like resolution refutation is superpolynomially larger than the size of a shortest DAG-like resolution refutation. An interesting future research is to find a set of formulas that separates tree-like and DAG-like resolutions in the rate of 2^{cn} for some constant c improving [2]. Another research topic is to find a tighter bound of the tree-like resolution for the pigeonhole principle. Note that an upper bound $O(n^2 n!)$ and a lower bound $(\frac{n}{4\log^2 n})^n$ obtained in this paper are tight in the sense that they both grow at the same rate of $n^{(1-o(1))n}$. An open question is whether we can get a tighter lower bound, e.g., $\Omega(n!)$.

Acknowledgments. The authors would like to thank Magnus M. Halldórsson for his valuable comments.

References

1. P. Beame and T. Pitassi, "Simplified and improved resolution lower bounds," *Proc. FOCS'96*, pp. 274–282, 1996.
2. M. L. Bonet, J. L. Esteban, N. Galesi and J. Johannsen, "Exponential separations between restricted resolution and cutting planes proof systems," *Proc. FOCS'98*, pp. 638–647, 1998.
3. S. Buss, "Polynomial size proofs of the propositional pigeonhole principle," *Journal of Symbolic Logic*, 52, pp. 916–927, 1987.
4. S. Buss and T. Pitassi, "Resolution and the weak pigeonhole principle," *Proc. CSL'97*, LNCS 1414, pp.149–156, 1997.
5. S. A. Cook and R. A. Reckhow, "The relative efficiency of propositional proof systems," *J. Symbolic Logic*, 44(1), pp. 36–50, 1979.
6. A. Haken, "The intractability of resolution," *Theoretical Computer Science*, 39, pp. 297–308, 1985.
7. P. Purdom, "A survey of average time analysis of satisfiability algorithms," *Journal of Information Processing*, 13(4), pp.449–455, 1990.
8. A. A. Razborov, A. Wigderson and A. Yao, "Read-Once Branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus," *Proc. STOC'97*, pp. 739–748, 1997.
9. A. Urquhart, "The complexity of propositional proofs," *The Bulletin of Symbolic Logic*, Vol. 1, No. 4, pp. 425-467, 1995.